Storage operators and forall-positive types of system TTR

Karim Nour

STORAGE OPERATORS and $\forall$-POSITIVE TYPES in TTR TYPE SYSTEM

Karim NOUR [1]
LAMA - Equipe de Logique, Université de Chambéry - 73376 Le Bourget du Lac cedex [2]

Abstract In 1990, J.L. Krivine introduced the notion of storage operator to simulate "call by value" in the "call by name" strategy. J.L. Krivine has shown that, using Gődel translation of classical into intuitionitic logic, we can find a simple type for the storage operators in $AF2$ type system. This paper studies the $\forall$-positive types (the universal second order quantifier appears positively in these types), and the Gődel transformations (a generalization of classical Gődel translation) of TTR type system. We generalize, by using syntaxical methods, the J.L. Krivine's Theorem about these types and for these transformations. We give a proof of this result in the case of the type of recursive integers.

Mathematics Subject Classification : 03B40, 68Q60
Keywords : Storage operator, Head normal form, Head reduction, $AF2$ type system, Least fixed point, TTR type system, Arrow type, Without-arrow type, $\forall$-positive type, $\perp$-type, Gődel transformation.

1 Introduction

The strategy of left reduction (iteration of head reduction denoted by $\succ$) has the following advantages :

- It has good mathematical properties stated by the normalisation Theorem : if a λ-term is normalizable, then we obtain the normal form by left reduction.
- It seems more economic since we compute a λ-term only when we need it.

Now, a drawback of the strategy of left reduction (call by name) is the fact that the argument of a function is computed as many times as it is used. The purpose of storage operators is precisely to correct this drawback.
Let F be a λ-term (a function), and $\underline{N}$ the set of normal Church integers. During the computation, by left reduction, of $(F)\theta_n$ (where $\theta_n \simeq_\beta \underline{n}$), θ_n may be computed several times (as many times as F uses it). We would like to transform $(F)\theta_n$ to $(F)\underline{n}$. We also want this transformation depends only on θ_n (and not F). In other words we look for some closed λ-terms T with the following properties :

[1] We thank R. David, J.L. Krivine, and M. Parigot for helpful discussions.
[2] e-mail nour@univ-savoie.fr

- For every F, $n \in \mathbb{N}$, and $\theta_n \simeq_\beta \underline{n}$, we have $(T)\theta_n F \succ (F)\underline{n}$;
- The computation time of the head reduction $(T)\theta_n F \succ (F)\underline{n}$ depends only on θ_n.

Therefore the first definition : A closed λ-term T is called storage operator for $\underline{N}$ if and only if for every $n \in \mathbb{N}$, and for every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\underline{n}$ (where f is a new variable).
It is clear that a storage operator satisfies the required properties. Indeed,

- Since we have $(T)\theta_n f \succ (f)\underline{n}$, then the variable f never comes in head position during the reduction, and we may then replace f by any λ-term.
- The computation time of the head reduction $(T)\theta_n F \succ (F)\underline{n}$ depends only on θ_n.

We showed (see [12]) that it is not possible to get the normal form of θ_n. We then change the definition : A closed λ-term T is called storage operator for $\underline{N}$ if and only if for every $n \in \mathbb{N}$, there is a closed λ-term $\tau_n \simeq_\beta \underline{n}$ (for example $\tau_n = (\underline{s})^n\underline{0}$, where $\underline{s}$ is a λ-term for the successor), such that for every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\tau_n$ (where f is a new variable).
If we take $T_1 = \lambda n((n)\lambda x\lambda y(x)\lambda z(y)(\underline{s})z)\lambda f(f)\underline{0}$, and $T_2 = \lambda n\lambda f(((n)\lambda x\lambda y(x)(\underline{s})y)f)\underline{0}$, then it is easy to check that : for every $\theta_n \simeq_\beta \underline{n}$, $(T_1)\theta_n f \succ (f)(\underline{s})^n\underline{0}$, and $(T_2)\theta_n f \succ (f)(\underline{s})^n\underline{0}$. Therefore T_1 and T_2 are storage operators for $\underline{N}$.

The $AF2$ type system is a way of interpreting the proof rules for the second order intuitionistic logic plus equational reasoning as construction rules for terms. In this system we have the possibility to define the data types, the representation in λ-calculus being automaticaly extracted from the logical definition of the data type. At the logical level the data type are defined by second order formulas expressing the usual iterative definition of the corresponding algebras of terms and the data receive the corresponding iterative definition in λ-calulus. For example, the type of integers is the formula : $N[x] = \forall X\{\forall y[X(y) \to X(sy)] \to [X(0) \to X(x)]\}$ (X is a unary predicate variable, 0 is a constant symbol for zero, and s is a unary function symbol for successor).
If we try to type a storage operator T in $AF2$ type system, we naturally find the type $\forall x\{N[x] \to [(N[x] \to O) \to O]\}$ (where O is a particular 0-ary predicate symbol which represents an arbitrary type). Indeed, if $\vdash_{AF2} \tau_n : N[s^n(0)]$, and f is of type $N[s^n(0)] \to O$, then $f : N[s^n(0)] \to O \vdash_{AF2} (f)\tau_n : O$. It is natural to have $(T)\theta_n f$ of type O. If $\vdash_{AF2} \theta_n : N[s^n(0)]$, then the type for T must be $\forall x\{N[x] \to [(N[x] \to O) \to O]\}$.
It is easy to check that $\vdash_{AF2} T_1, T_2 : \forall x\{N[x] \to [(N[x] \to O) \to O]\}$.
The type $\forall x\{N[x] \to [(N[x] \to O) \to O]\}$ does not characterize the storage operators. Indeed, if we take $T = \lambda n\lambda f(f)n$, we obtain :

- $n : N[x], f : N[x] \to O \vdash_{AF2} (f)n : O$, then, $\vdash_{AF2} T : \forall x\{N[x] \to [(N[x] \to O) \to O]\}$.
- For every $\theta_n \simeq_\beta \underline{n}$, $(T)\theta_n f \succ (f)\theta_n$, therefore T is not a storage operator for $\underline{N}$.

This comes from the fact that the type $\forall x\{N[x] \to [(N[x] \to O) \to O]\}$ does not take into account the independance of τ_n with θ_n. To solve this problem, we must prevent the use of the first $N[x]$ in $\forall x\{N[x] \to [(N[x] \to O) \to O]\}$ as well as his subtypes to prove the second. Therefore, we will replace the first $N[x]$ by a new type $N^*[x]$ with the following properties :

- $\vdash_{AF2} \underline{n} : N^*[s^n(0)]$ (for example, take $N^*[x] = \forall X\{\forall y[F(X,y) \to F(X,sy)] \to [F(X,0) \to F(X,x)]\})$;
- If $\nu : N^*[x], x_i : \forall y[F(G,y) \to F(G,sy)], y_j : F(H,a) \vdash_{AF2} t : N[s^n(0)]$, then $\vdash_{AF2} t' : N[s^n(0)]$, where t' is the normal form of t ;
- There is a closed λ-term T, such that $\vdash_{AF2} T : \forall x\{N^*[x] \to [(N[x] \to O) \to O]\}$.

A simple solution for the second property is to take a formula $F(X,a)$ ending with a new constant symbol. Indeed, since $N[x]$ does not contain this symbol, we cannot use the variables ν, x_i, y_j in the typing of t'. We suggest the following proposition :

$$N^*[x] = \forall X\{\forall y[(X(y) \to O) \to (X(sy) \to O)] \to [(X(0) \to O) \to (X(x) \to O)]\}.$$

It is easy to chech that $\vdash_{AF2} T_1, T_2 : \forall x\{N^*[x] \to [(N[x] \to O) \to O]\}$ (see [6] and [12]).
For each formula F of $AF2$, we indicate by F^* the formula obtained by putting $\neg$ in front of each atomic formulas of F (F^* is called the Gődel translation of F).
J.L. Krivine has shown that the type $\forall x\{N^*[x] \to \neg\neg N[x]\}$ characterize the storage operators for $\underline{N}$ (see [6]). But the λ-term τ_n obtained may contain variables substituted by λ-terms $u_1, ..., u_m$ depending on θ_n. Since the λ-term τ_n is $\beta\eta$-equivalent to $\underline{n}$, therefore, the left reduction of the $\tau_n[u_1/x_1, ..., u_m/x_m]$ is equivalent to the left reduction of τ_n and the λ-terms $u_1, ..., u_m$ will therefore never be evaluated during the reduction.
Taking into account the above remarks, we modify again the definition : A closed λ-term T is called a storage operator for $\underline{N}$ if and only if for every $n \in \mathbb{N}$, there is a λ-term $\tau_n \simeq_\beta \underline{n}$, such that for every $\theta_n \simeq_\beta \underline{n}$, there is a substitution σ, such that $(T)\theta_n f \succ (f)\sigma(\tau_n)$ (where f is a new variable).

The $AF2$ type system is satisfactory from an extensional point of view : one can construct programs for all the functions whose termination is provable in the second order Peano arithmetic. But from an intensional point of view the situation is very different : we cannot always obtain the simple (in term of time complexity, for instance) programs we need. For example we cannot find a λ-term of type $\forall x \forall y\{N[x], N[y] \to N[min(x,y)]\}$ (min is a binary function symbol defined by equations) in $AF2$ type system that computes the minimum of two Church integers in time $O(min)$ [3]

[3] R. David gives a λ-term of type $N, N \to N$ ($N = \forall X\{[X \to X] \to [X \to X]\}$) in F type system that computes the minimum of two Church integers in time $O(min.Log(min))$. The notion of storage operators plays an important tool in this constraction (see [2]).

The TTR type system is an extension of $AF2$ based on recursive definitions of types, which is intented to solve the basic problems of efficiency mentioned before. In TTR we have a logical operator μ of least fixed point. If A is a formula, C an n-ary predicate symbol which appears and occurs positively in A, $x_1, ..., x_n$ first order variables, and $t_1, ..., t_n$ terms, then $\mu C x_1 ... x_n A < t_1, .., t_n >$ is a formula called the least fixed point of A in C calculated over the terms $t_1, ..., t_n$. The interded logical meaning of the formula $\mu C x_1 ... x_n A < t_1, .., t_n >$ is $K(t_1, ..., t_n)$, where K is the least X, such that $X(x_1, ..., x_n) \longleftrightarrow A$. TTR allows to define the multisorted term algebras as least fixed points. For example the type of recursive integers is the formula : $N^r[x] = \mu C z[\forall X\{\forall y[C(y) \to X(sy)] \to [X(0) \to X(z)]\}] < x >$ (X is a unary predicate variable, 0 is a constant symbol for zero, and s is a unary function symbol for successor).

In this paper we study the types D of TTR, and the transformations *, for which we have the following result : if $\vdash_{TTR} T : D^* \to \neg\neg D$, then for every λ-term t with $\vdash_{TTR} t : D$, there are λ-terms τ_t and τ'_t such that $\tau_t \simeq_\beta \tau'_t$, $\vdash_{TTR} \tau'_t : D$, and for every $\theta_t \simeq_\beta t$, there is a substitution σ, such that $(T)\theta_t f \succ (f)\sigma(\tau_t)$ (where f is a new variable).
We prove [4] that, to obtain this result, it suffies to assume that :

- The universal second order quantifier appears positively in D ($\forall$-positive type) [5].
- The transformation * satisfies the following properties :

  - If $A = C(t_1, ..., t_n)$, then $A^* = A$;

  - If $A = X(t_1, ..., t_n)$, then $A^* = F_X[t_1/x_1, ..., t_n/x_n] < X_1, ..., X_r >$ where F_X is a formula ending with $\perp$ and having $x_1, ..., x_n, X_1, ..., X_r$ as free variables ;

  - $(A \to B)^* = A^* \to B^*$;

  - $(\forall x A)^* = \forall x A^*$.

  - $(\forall X A)^* = \forall X_1 ... X_r A^*$.

  - $(\mu C x_1 ... x_n A < t_1, .., t_n >)^* = \mu C x_1 ... x_n A^* < t_1, .., t_n >$.

We give the proof of this result in the case of the type of recurcive integers.

2 Basic notions of pure λ-calculus

Our notation is standard (see [1] and [5]).
We denote by Λ the set of terms of pure λ-calculus, also called λ-terms.
Let $t, u, u_1, ..., u_n \in \Lambda$, the application of t to u is denoted by $(t)u$. In the same way we write

[4] J.L. Krivine and the author proved independely the same result for $AF2$ type system (see [7] and [12]).
[5] This types were studied by some authors (in particular R. Labib-Sami), and have remarkable properties (see [8]).

$(t)u_1...u_n$ instead of $(...((t)u_1)...)u_n$.
The β-reduction (resp. β-equivalence) is denoted by $t \rightarrow_\beta u$ (resp. $t \simeq_\beta u$).
The set of free variables of a λ-term t is denoted by $Fv(t)$.
The notation $t[u_1/x_1, ..., u_n/x_n]$ represents the result of the simultaneous substitution of λ-terms $u_1, ..., u_n$ to the free variables $x_1, ..., x_n$ of t (after a suitable renaming of the bounded variables of t).
With each normal λ-term, we associate a set of λ-terms $STE(t)$ by induction :
if $t = \lambda x_1...\lambda x_n(y)t_1...t_m$, then $STE(t) = \{t\} \bigcup \bigcup_{1 \leq i \leq m} STE(t_i)$.
Let us recall that a λ-term t either has a head redex [i.e. $t = \lambda x_1...\lambda x_n(\lambda xu)vv_1...v_m$, the head redex being $(\lambda xu)v$], or is in head normal form [i.e. $t = \lambda x_1...\lambda x_n(x)vv_1...v_m$].
The notation $t \succ t'$ means that t' is obtained from t by some head reductions, and we denote by $n(t, t')$, the number of steps to go from t to t'.
A λ-term t is said to be solvable if and only if the head reduction of t terminates.
We define an equivalence relation $\sim$ on Λ by : $u \sim v$ if and only if there is a t, such that $u \succ t$, and $v \succ t$. In particular, if v is in head normal form, then $u \sim v$ means that v is the head normal form of u.

Theorem 2.1 *([6]). If $t \succ t'$, then for every $u_1, ..., u_n \in \Lambda$:*
1) there is a $v \in \Lambda$, such that $(t)u_1...u_n \succ v$, $(t')u_1...u_n \succ v$, and $n((t)u_1...u_n, v) = n((t')u_1....u_n, v) + n(t, t')$.
2) $t[u_1/x_1, ..., u_n/x_n] \succ t'[u_1/x_1, ..., u_n/x_n]$, and $n(t[u_1/x_1, ..., u_n/x_n], t'[u_1/x_1, ..., u_n/x_n]) = n(t, t')$.

Remark. Theorem 2.1 shows that to make the head reduction of $(t)u_1...u_n$ (resp. $t[u_1/x_1, ..., u_n/x_n]$), it is equivalent (same result, and same number of steps) to make some steps in the head reduction of t, and then make the head reduction of $(t')u_1...u_n$ (resp. $t'[u_1/x_1, ..., u_n/x_n]$).

3 Basic notions of typed λ-calculus

3.1 The $AF2$ type system

The types will be formulas of second order predicate logic over a given language.
The logical symbols are $\perp$ (for absurd), $\rightarrow$ and $\forall$ (and no other ones).
There are individual variables : $x, y, ...$ (also called first order variables) and n-ary predicate variables ($n = 0, 1, ...$) : $X, Y, ...$ (also called second order variables).
The terms and the formulas are up in the usual way.
The formula $F_1 \rightarrow (F_2 \rightarrow (... \rightarrow (F_n \rightarrow G)...))$ is denoted by $F_1, F_2, ..., F_n \rightarrow G$, and $F \rightarrow \perp$ is denoted by $\neg F$. The formula $\forall v_1...\forall v_n F$ is denoted by $\forall \mathbf{v} F$, and the sentence "$\mathbf{v}$ is not free in A" means that for all $1 \leq i \leq n$, v_i is not free in A.

If X is a unary predicate variable, t and t' two terms, then the formula $\forall X[Xt \rightarrow Xt']$ is denoted by $t = t'$, and is said to be equation. A particular case of $t = t'$ is a formula of the forme $t[u_1/x_1, ..., u_n/x_n] = t'[u_1/x_1, ..., u_n/x_n]$ or $t'[u_1/x_1, ..., u_n/x_n] = t[u_1/x_1, ..., u_n/x_n]$, $u_1, ..., u_n$ being terms of the language.
After, we denote by **E** a system of function equations.
A context Γ is a set of the form $x_1 : A_1, ..., x_n : A_n$ where $x_1, ..., x_n$ are distinct variables and $A_1, ..., A_n$ are formulas.
We are going to describe a system of typed λ-calculus called second order functional arithmetic (shortened in $AF2$ for Arithmétique Fonctionnelle du seconde ordre). The rules of typing are the following :

(1) $\Gamma, x : A \vdash_{AF2} x : A$.

(2) If $\Gamma, x : B \vdash_{AF2} t : C$, then $\Gamma \vdash_{AF2} \lambda xt : B \rightarrow C$.

(3) If $\Gamma \vdash_{AF2} u : B \rightarrow C$, and $\Gamma \vdash_{AF2} v : B$, then $\Gamma \vdash_{AF2} (u)v : C$.

(4) If $\Gamma \vdash_{AF2} t : A$, and x does not appear in Γ, then $\Gamma \vdash_{AF2} t : \forall xA$.

(5) If $\Gamma \vdash_{AF2} t : \forall xA$, then, for every term u, $\Gamma \vdash_{AF2} t : A[u/x]$.

(6) If $\Gamma \vdash_{AF2} t : A$, and X does not appear in Γ, then $\Gamma \vdash_{AF2} t : \forall XA$.

(7) If $\Gamma \vdash_{AF2} t : \forall XA$, then, for every formula G, $\Gamma \vdash_{AF2} t : A[G/X(x_1, ..., x_n)]$ (*)

(8) If $\Gamma \vdash_{AF2} t : A[u/x]$, then $\Gamma \vdash_{AF2} t : A[v/x]$, $u = v$ being a particular case of an equation of **E** .

(*) $A[G/X(x_1, ..., x_n)]$ is obtained by replacing in A each atomic formula $X(t_1, ..., t_n)$ by $G[t_1/x_1, ..., t_n/x_n]$. To simplify, we write sometimes $A[G/X]$ instead of $A[G/X(x_1, ..., x_n)]$.

Whenever we obtain the typing $\Gamma \vdash_{AF2} t : A$ by means of these rules, we say that "the λ-term t is of type A in the context Γ, with respect to the equation of **E** ".

Theorem 3.1 *([5],[9]).*
1) Conservation Theorem: If $\Gamma \vdash_{AF2} t : A$, and $t \rightarrow_\beta t'$, then $\Gamma \vdash_{AF2} t' : A$.
2) Strong normalization: If $\Gamma \vdash_{AF2} t : A$, then t is strongly normalizable.

3.2 The TTR type system

Let X be a predicate variable or predicate symbol, and A a type of $AF2$.
We define the notions "X is positive in A" and "X is negative in A" by induction :

- If X does not appears in A, then X is positive and negative in A ;

- If $A = X(t_1, ..., t_n)$, then X is positive in A, and X is not negative in A ;

- If $A = B \to C$, then X is positive (resp. negative) in A if and only if X is negative (resp. positive) in B, and X is positive (resp. negative) in C ;

- If $A = \forall v B$, and $v \neq X$, then X is positive (resp. negative) in A if and only if X is positive (resp. negative) in B.

We add to the second order predicate calculus a new logic symbol μ, and we allow a new construction for formulas : if A is a formula, C an n-ary predicate symbol which appears positively in A, $x_1, ..., x_n$ first order variables, and $t_1, ..., t_n$ terms, then $\mu C x_1 ... x_n A < t_1, ..., t_n >$ is a formula called the least fixed point of A in C calculated over the terms $t_1, ..., t_n$.
We extend the notions "X is positive in a type" and "X is negative in a type" by the following way : X is positive (resp. negative) in $\mu C x_1 ... x_n A < t_1, ..., t_n >$ if and only if X is positive (resp. negative) in A.
We extend the definition of the substitution by assuming that $C, x_1, ..., x_n$ are bounded in the formula $\mu C x_1 ... x_n A < t_1, ..., t_n >$.
We define on these formulas a binary relation $\subseteq$ by : $A \subseteq B$ if and only if it is obtained by using the following rules :

$(ax) A \subseteq A$ $\qquad$ $(\to) \dfrac{A \subseteq A' \quad B \subseteq B'}{A' \to B \subseteq A \to B'}$

$(\forall i_g) \dfrac{A[G/v] \subseteq B}{\forall v A \subseteq B}$ (1) $\qquad$ $(\forall i_d) \dfrac{A \subseteq B}{A \subseteq \forall v B}$ (2)

$(e) \dfrac{A \subseteq B[v/y]}{A \subseteq B[w/y]}$ (3) $\qquad$ $(tr) \dfrac{A \subseteq D \quad D \subseteq B}{A \subseteq B}$

(μ_d) $D[\mu C x_1 ... x_m D < z_1, ..., z_m > / C(z_1, ..., z_m)][t_1/x_1, ..., t_m/x_m] \subseteq \mu C x_1 ... x_m D < t_1, ..., t_m >$

(μ'_g) $\mu C x_1 ... x_m D < t_1, ..., t_m > \subseteq D[\mu C x_1 ... x_m D < z_1, ..., z_m > / C(z_1, ..., z_m)][t_1/x_1, ..., t_m/x_m]$

$$(\mu_g) \ \frac{D[E/C(x_1, ..., x_m)] \subseteq E}{\mu C x_1 ... x_m D < t_1, ..., t_m > \subseteq E[t_1/x_1, ..., t_m/x_m]}$$

(1) G is a formula if v is a second order variable, and a term if v is a first order variable.
(2) v is not free in A.
(3) $v = w$ is a particular case of an equation of $\mathbf{E}$.

(μ_d) and (μ'_g) are the rules of factorisation and development of a fixed point.
(μ_g) expresses the fact that $\mu C x_1 ... x_m D < t_1, ..., t_m >$ is a least fixed point.

We are going to describe a system of typed λ-calculus called theory of recursive types (shortened in TTR for Théorie des Types Récursifs) where the types are formulas of language. The rules of typing are the following :

- The typing rules (1),...,(8) of $AF2$ type system.

- $(\subseteq)$ $\dfrac{\Gamma \vdash_{TTR} t : A \quad A \subseteq B}{\Gamma \vdash_{TTR} t : B}$

- (Y) $\dfrac{\Gamma \vdash_{TTR} t : \forall x_1 ... \forall x_m [C(x_1, ..., x_m) \to E] \to \forall x_1 ... \forall x_m [D \to E]}{\Gamma \vdash_{TTR} (Y)t : \forall x_1 ... \forall x_m [\mu C x_1 ... x_m D < x_1, ..., x_m > \to E]}$
where C is not free in E and G, and Y is the Turing's fixed point.

The rule (Y) expresses also the fact that $\mu C x_1 ... x_m D < t_1, ..., t_m >$ is a least fixed point.

Theorem 3.2 *([12],[18]).*
1) Conservation Theorem If $\Gamma \vdash_{TTR} t : A$, *and* $t \to_\beta t'$, *then* $\Gamma \vdash_{TTR} t' : A$.
2) Strong normalization If $\Gamma \vdash_{TTR} t : A$ *without using the rule* (Y), *then* t *is strongly normalizable.*
3) Weak normalization If $\Gamma \vdash_{TTR} t : A$, *and if all least fixed points of* A *are positives, then* t *is normalizable.*

The $TTR^\diamond$ type system is the subsystem of TTR where we only have propositional variables and constants (predicate variables or predicate symbols are of arity 0). So, first order variables, function symbols, and finite sets of equations are useless. With each predicate variable (resp. predicate symbol) X, we associate a predicate variable (resp. a predicate symbol) $X^\diamond$ of $TTR^\diamond$ type system. For every formula A of TTR, we define the formula $A^\diamond$ of $TTR^\diamond$ obtained by forgetting in A the first order part. If $\Gamma = x_1 : A_1, ..., x_n : A_n$ is a context of TTR, then we denote by $\Gamma^\diamond$, the context $x_1 : A_1^\diamond, ..., x_n : A_n^\diamond$ of $TTR^\diamond$. We write $\Gamma \vdash_{TTR^\diamond} t : A$ if t is tyable in $TTR^\diamond$ of type A in the context Γ.

Theorem 3.3 *If* $\Gamma \vdash_{TTR} t : A$, *then* $\Gamma^\diamond \vdash_{TTR^\diamond} t : A^\diamond$.

Proof By induction on the length of the derivation $\Gamma \vdash_{TTR} t : A$. □

Theorem 3.4
1) Conservation Theorem If $\Gamma \vdash_{TTR^\diamond} t : A$, *and* $t \to_\beta t'$, *then* $\Gamma \vdash_{TTR^\diamond} t' : A$.
2) Strong normalization If $\Gamma \vdash_{TTR^\diamond} t : A$ *without using the rule* (Y), *then* t *is strongly normalizable.*
3) Weak normalization If $\Gamma \vdash_{TTR^\diamond} t : A$, *and if all least fixed points of* A *are positives, then* t *is normalizable.*

Proof We use Theorems 3.2 and 3.3. □

Remark We cannot if the reverse of 2)-Theorem 3.2 is true, but the λ-term $t = \lambda x(\lambda y((x)(y)\lambda xx)(y)\lambda x\lambda yx)\lambda x(x)x$ (which is strongly normalizable, and untypable in $AF2$ type system (see [3])) is typable in TTR type system. Indeed, if we take $B = \mu C(\forall XX \to C)$, we check easily that $\vdash_{TTR^\diamond} t : [B \to (B \to B)] \to B$.

4 Properties of TTR type system

4.1 Permutations Lemmas

Lemma 4.1 *1) The typing rules (5), (7), and (8) are admissible.*
2) In the typing, we may replace the succession of n times ($\subseteq$) and m times (4) (resp. (6)), by the succession of m times (4) (resp. (6)) and n times ($\subseteq$).
3) If $\Gamma \vdash_{TTR} t : B$ is derived from $\Gamma \vdash_{TTR} t : A$, then we may assume that we begin by the applications of (4), (6), and next by ($\subseteq$).

Proof Easy. □

Lemma 4.2 *1) If $A \subseteq B$, then, for every sequence of terms and/or formulas $\mathbf{G}$, $A[\mathbf{G}/\mathbf{v}] \subseteq B[\mathbf{G}/\mathbf{v}]$, and we use the same proof rules.*
2) If $\Gamma \vdash_{TTR} t : A$, then, for every sequence of terms and/or formulas $\mathbf{G}$, $\Gamma[\mathbf{G}/\mathbf{v}] \vdash_{TTR} t : A[\mathbf{G}/\mathbf{v}]$, and we use the same typing rules.

Proof By induction on the length of the derivation $A \subseteq B$ (resp. $\Gamma \vdash_{TTR} t : A$). □

Corollary 4.1 *If $\Gamma, x : A \vdash_{TTR} (x)u_1...u_n : B$, then :*
$n = 0$, and there is $\mathbf{v_0}$ not free in A and Γ, such that $\forall \mathbf{v_0} A \subseteq B$,
or
$n \geq 1$, and there are types C_i,B_i $(i = 1, ..., n)$ and $\mathbf{v_i}(i = 1, n)$ not free in A and Γ, such that $\forall \mathbf{v_0} A \subseteq C_1 \to B_1$, $\forall \mathbf{v_i} B_i \subseteq C_{i+1} \to B_{i+1}$ $1 \leq i \leq n-1$, $\forall \mathbf{v_n} B_n \subseteq B$, and $\Gamma, x : A \vdash_{TTR} u_i : C_i$ $1 \leq i \leq n$.

Proof By induction on n. □

Lemma 4.3 *1) If X is positive (resp. negative) in D, and $A \subseteq B$, then $D[A/X] \subseteq D[B/X]$ (resp. $D[B/X] \subseteq D[A/X]$).*
2) We may eliminate the rule (μ'_g).

Proof 1) By induction on D.
2) By rule (μ_d), we have $A[\mu C x_1...x_n A < y_1, ..., y_n > /C(y_1, ..., y_n)] \subseteq \mu C x_1...x_n A < x_1, ..., x_n >$,

then, by 1), $A[A[\mu Cx_1...x_nA < y_1,...,y_n > /C(y_1,...,y_n)]/C(x_1,...,x_n)] \subseteq A[\mu Cx_1...x_nA < x_1,...,x_n > /C(x_1,...,x_n)]$, and, by using the rule (μ_g), we obtain $\mu Cx_1...x_nA < t_1,...,t_n > \subseteq A[\mu Cx_1...x_nA < y_1,...,y_n > /C(y_1,...,y_n)][t_1/x_1,...,t_n/x_n]$. □

4.2 Without-arrow types and arrow types

Definitions
1) A type A is said to be without-arrow type if and only if A does not contain any arrow.
2) Each without-arrow type A contains a unique atomic formula $X(t_1,...,t_n)$. We denote X by $At(A)$. We distinguish between two kinds of without-arrow types :
- A without-arrow type A is said to be of kind 1 if and only if $At(A)$ is free in A.
- A without-arrow type A is said to be of kind 2 if and only if $At(A)$ is bounded in A.

Lemma 4.4 *1) If A is a without-arrow type of kind 1, and $A \subseteq B$, then B is a without-arrow type of kind 1, and $At(A) = At(B)$.*
2) If A is a without-arrow type of kind 2, then, for every type B, we have $A \subseteq B$.

Proof 1) By induction on the length of the derivation $A \subseteq B$.
2) Easy. □

Definition A type A is said to be arrow type if and only if A contains at least an arrow.

Lemma 4.5 *If A is an arrow type, and $A \subseteq B$, then B is an arrow type.*

Proof By induction on the length of the derivation $A \subseteq B$. □

Corollary 4.2 *Let A be an atomic formula. If $\Gamma \vdash_{TTR} t : A$, then t does not begin by λ. Other words, if $\Gamma \vdash_{TTR} \lambda xu : B$, then B is an arrow type.*

Proof If t begins by λ, then there are E, F, and $\mathbf{v}$, such that $\forall \mathbf{v}(E \to F) \subseteq A$, therefore, by Lemma 4.5, A is an arrow type. □

Definition For every arrow type A, we define the type $Rep(A)$ as follows, by induction on A :

- $Rep(E \to F) = E \to F$;

- $Rep(\forall vB) = \forall v Rep(B)$;

- $Rep(\mu Cx_1...x_nB < t_1,...,t_n >) =$
$Rep(B)[\mu Cx_1...x_nB < y_1,...,y_n > /C(y_1,..,y_n)][t_1/x_1,...,t_n/x_n]$.

Lemma 4.6 *If A is an arrow type, then :*
1) there are G, D and $\mathbf{v}$ such that $Rep(A) = \forall \mathbf{v}(G \to D)$.
2) $A \subseteq Rep(A)$, and $Rep(A) \subseteq A$.

Proof By induction on A. □

Remark. The Lemma 4.6 means that if A is an arrow type, then $Rep(A)$ is an "equivalent" type to A of the form $\forall \mathbf{v}(G \to D)$. In the rest of the paper, we denoted G by A_g and D by A_d.

Lemma 4.7 *Let A, B be two types, and X, X' two predicate variables or predicate symbols, such that X' is not free in A.*
1) If X is positive in A, and X' is positive in B, then X' is positive in $A[B/X]$.
2) If X is positive in A, and X' is negative in B, then X' is negative in $A[B/X]$.
3) If X is negative in A, and X' is positive in B, then X' is negative in $A[B/X]$.
4) If X is negative in A, and X' is negative in B, then X' is positive in $A[B/X]$.

Proof By induction on A. □

Lemma 4.8 *Let A be an arrow type.*
1) If X is positive (resp. negative) in A, then X is positive (resp. negative) in $Rep(A)$.
2) If $\mathbf{G}$ is a sequence of terms and/or formulas, then $Rep(A[\mathbf{G}/\mathbf{v}]) = Rep(A)[\mathbf{G}/\mathbf{v}]$.

Proof 1) We argue by induction on A. The only non-trivial case is the one where $A = \mu C x_1 ... x_n B < t_1, ..., t_n >$. If X is positive (resp. negative) in A, then X is positive (resp. negative) in B. By the induction hypothesis, we have X is positive (resp. negative) in $Rep(B)$, therefore, by Lemma 4.7, X is positive (resp. negative) in $Rep(A)$.
2) By induction on A. □

Theorem 4.1 *Let A, B be two arrow types, such that $Rep(A) = \forall \mathbf{v}(A_g \to A_d)$ and $Rep(B) = \forall \mathbf{v}'(B_g \to B_d)$. If $A \subseteq B$, then there is a sequence of terms and/or formulas $\mathbf{G}$, such that $B_g \subseteq A_g[\mathbf{G}/\mathbf{v}]$, and $A_d[\mathbf{G}/\mathbf{v}] \subseteq B_d$.*

Proof We argue by induction on the length of the derivation $A \subseteq B$. Let us look at the rule used in the last step. The only non-trivial cases are :

- (tr) : then $A \subseteq D$, and $D \subseteq B$. If $Rep(D) = \forall \mathbf{v}"(D_g \to D_d)$, by the induction hypothesis, there are sequences $\mathbf{G}$ and $\mathbf{G}"$ such that $D_g \subseteq A_g[\mathbf{G}/\mathbf{v}]$, $A_d[\mathbf{G}/\mathbf{v}] \subseteq D_d$, $B_g \subseteq D_g[\mathbf{G}"/\mathbf{v}"]$, and $D_d[\mathbf{G}"/\mathbf{v}"] \subseteq B_d$. It is clear that we may assume that $\mathbf{v}"$ is not free in A_g and A_d, therefore, by Lemma 4.2, we have $B_g \subseteq A_g[\mathbf{G}/\mathbf{v}][\mathbf{G}"/\mathbf{v}"]$, and $A_d[\mathbf{G}/\mathbf{v}][\mathbf{G}"/\mathbf{v}"] \subseteq B_d$. Let $\mathbf{G}' = \mathbf{G}[\mathbf{G}"/\mathbf{v}"]$, then $B_g \subseteq A_g[\mathbf{G}'/\mathbf{v}]$, and $A_d[\mathbf{G}'/\mathbf{v}] \subseteq B_d$.

- (μ_d) : then $A = D[\mu C x_1 ... x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][t_1/x_1, .., t_k/x_k]$, and $B = \mu C x_1 ... x_k D < t_1, ..., t_k >$. Therefore, by Lemma 4.8, $Rep(A) = Rep(B)$, $A_g = B_g$, and $B_d = A_d$, and so $B_g \subseteq A_g$, and $A_d \subseteq B_d$.

- (μ_g) : then $A = \mu C x_1...x_k D < t_1, ..., t_k >$, $B = E[t_1/x_1, ..., t_k/x_k]$, and $D[E/C(x_1, ..., x_k)] \subseteq E$. Therefore $Rep(D) = \forall \mathbf{v}(D_g \to D_d)$ with
$D_g[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][t_1/x_1, .., t_k/x_k] = A_g$,
$D_d[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][t_1/x_1, .., t_k/x_k] = A_d$, and
$Rep(E) = \forall \mathbf{v}'(E_g \to E_d)$ with $E_g[t_1/x_1, ..., t_k/x_k] = Bg$, $E_d[t_1/x_1, ..., t_k/x_k] = B_d$.
By the induction hypothesis, there is a sequence $\mathbf{G}$**,** such that $E_g \subseteq D_g[E/C(x_1, ..., x_k)][\mathbf{G}/\mathbf{v}]$, and $D_d[E/C(x_1, ..., x_k)][\mathbf{G}/\mathbf{v}] \subseteq E_d$. C is positive in D, therefore, by Lemma 4.8, C is negative in D_g, and C is positive in D_d.
$D[E/C(x_1, ..., x_k)] \subseteq E$, then $\mu C x_1...x_k D < y_1, ..., y_k > \subseteq E[y_1/x_1, ..., y_k/x_k]$, and, by 1)-Lemma 4.3, $Eg \subseteq Dg[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][\mathbf{G}/\mathbf{v}]$, and
$D_d[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][\mathbf{G}/\mathbf{v}] \subseteq E_d$, and so, by Lemma 4.2,
$E_g[t_1/x_1, ..., t_k/x_k] \subseteq D_g[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][\mathbf{G}/\mathbf{v}][t_1/x_1, ..., t_k/x_k]$,
and $D_d[\mu C x_1...x_k D < y_1, ..., y_k > /C(y_1, ..., y_k)][\mathbf{G}/\mathbf{v}][t_1/x_1, ..., t_n/x_n] \subseteq E_d[t_1/x_1, ..., t_k/x_k]$.
Let $\mathbf{G}' = \mathbf{G}[t_1/x_1, ..., t_k/x_k]$, then $B_g \subseteq A_g[\mathbf{G}'/\mathbf{v}]$,*and* $A_d[\mathbf{G}'/\mathbf{v}] \subseteq B_d$. □

Corollary 4.3 *Let B be an atomic formula. If $\Gamma, x : A \to B \vdash_{TTR} (x)u_1...u_n : C$, then $n = 1$, and $\Gamma, x : A \to B \vdash_{TTR} u_1 : A$.*

Proof By Corollary 4.1, we have $\forall \mathbf{v}(A \to B) \subseteq F \to G$, $\Gamma, x : A \to B \vdash_{TTR} u_1 : F$, and $\mathbf{v}$ is not free in Γ and $A \to B$. Therefore, by Theorem 4.1, $F \subseteq A$, and $B \subseteq G$, then $\Gamma, x : A \to B \vdash_{TTR} u_1 : A$. If $n > 1$, then $\forall \mathbf{v}' G \subseteq H \to J$, and $\mathbf{v}'$ is not free in Γ and $A \to B$. Therefore $\forall \mathbf{v}' B \subseteq H \to J$, and $\forall \mathbf{v}' B$ is a without-arrow type of kind 1. A contradiction. □

Lemma 4.9 *If $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} t : A$, $B_i \subseteq A_i$ $1 \leq i \leq n$, and $A \subseteq B$, then $x_1 : B_1, ..., x_n : B_n \vdash_{TTR} t : B$.*

Proof We argue by induction on t. The only non-trivial cases are :

- If $t = \lambda x u$, then $x_1 : A_1, ..., x_n : A_n, x : E \vdash_{TTR} u : F$, $\forall \mathbf{v}(E \to F) \subseteq A$, and $\mathbf{v}$ is not free in E and A_j $1 \leq j \leq n$. We may assume that $\mathbf{v}$ is not free in E and B_j $1 \leq j \leq n$. By the induction hypothesis, we have $x_1 : B_1, ..., x_n : B_n, x : E \vdash_{TTR} u : F$, and so $x_1 : B_1, ..., x_n : B_n \vdash_{TTR} t : B$.

- If $t = (Y)u$, then $\forall \mathbf{v} \forall y_1...\forall y_m[\mu C y_1...y_m E < y_1, ..., y_m > \to D]) \subseteq A$, $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} u : \forall y_1...\forall y_m[C(y_1, ..., y_m) \to D] \to \forall y_1...\forall y_m[E \to D]$, C is positive in E, C is not free in D, and $\mathbf{v}$ is not free in A_j $1 \leq j \leq n$. We may assume that $\mathbf{v}$**,** C are not free in B_j $1 \leq j \leq n$. By the induction hypothesis, we have $x_1 : B_1, ..., x_n : B_n \vdash_{TTR} u : \forall y_1..\forall y_m[C(y_1, ..., y_m) \to D] \to \forall y_1...\forall y_m[E \to D]$, and so $x_1 : B_1, ..., x_n : B_n \vdash_{TTR} (Y)u : A$. □

5 ∀-positive types

5.1 Properties of ∀-positive types

Definition We define two sets of types, the set Ω^+ of ∀-positive types, and the set Ω^- of ∀-negative types in the following way :

- If A is an atomic type, then $A \in \Omega^+$, and $A \in \Omega^-$;

- If $T^+ \in \Omega^+$, and $T^- \in \Omega^-$, then, $T^- \to T^+ \in \Omega^+$, and $T^+ \to T^- \in \Omega^-$;

- If $T^+ \in \Omega^+$, then $\forall x T^+ \in \Omega^+$;

- If $T^+ \in \Omega^+$, then $\forall X T^+ \in \Omega^+$;

- If $T^- \in \Omega^-$, then $\forall x T^- \in \Omega^-$;

- If $T^- \in \Omega^-$, and X is not free in T^-, then $\forall X T^- \in \Omega^-$- ;

- If $T^+ \in \Omega^+$, $x_1, ..., x_n$ first order variables, $t_1, ..., t_n$ terms, C an n-ary predicate symbol which appears and is positive in T^+, then $\mu C x_1 ... x_n T^+ < t_1, ..., t_n > \in \Omega^+$.

Remarks
1) A least fixed point is not a ∀-negative type.
2) If $T^+ \in \Omega^+$, then all least fixed points of T^+ are positives. Therefore, by 3)-Theorem 3.2, if $\Gamma \vdash_{TTR} t : T^+$, then t is normalizable.

Lemma 5.1 *Let $T^-, T'^- \in \Omega^-$, $T^+, T'^+ \in \Omega^+$, and X a predicate variable or predicate symbol.*
1) If X is positive (resp. negative) in T^-, then $T^-[T'^-/X] \in \Omega^-$ (resp. $T^-[T'^+/X] \in \Omega^-$).
2) If X is positive (resp. negative) in T^+, then $T^+[T'^+/X] \in \Omega^+$ (resp. $T^+[T'^-/X] \in \Omega^+$).
3) If $T[F/X] \in \Omega^+$ (resp. $T[F/X] \in \Omega^-$), then $T \in \Omega^+$ (resp. $T \in \Omega^-$).

Proof 1), 2) By induction on T^- and T^+.
3) By induction on T. □

Definition With each type T of TTR, we associte the set $Fv_2(T)$ of free predicate variables and free predicate symbols of T.

Theorem 5.1 *Let $T^- \in \Omega^-$, and $T^+ \in \Omega^+$.*
1) If $T^- \subseteq A$, then $A \in \Omega^-$, and $Fv_2(A) \subseteq Fv_2(T^-)$.
2) If $B \subseteq T^+$, then $B \in \Omega^+$, and $Fv_2(B) \subseteq Fv_2(T^+)$.

Proof We argue by induction on the length of the derivations $T^- \subseteq A$, and $B \subseteq T^+$. Let us look at the rule used in the last step.

1) The only non-trivial case is (μ_d).
Then $T^- = T'[\mu C x_1...x_n T' < y_1, ..., y_n > /C(y_1, ..., y_n)][t_1/x_1, ..., t_n/x_n]$, and
$A = \mu C x_1...x_n T' < t_1, .., t_n >$. Since $T^- \in \Omega^-$, then, by Lemma 5.1, $\mu C x_1...x_n T' < y_1, ..., y_n > \in \Omega^-$, which is impossible.
2) The only non-trivial cases are :

- (μ_d) : then $B = D[\mu C x_1...x_n D < y_1, ..., y_n > /C(y_1, ..., y_n)][t_1/x_1, ..., t_n/x_n]$, and $T^+ = \mu C x_1...x_n D < t_1, ..., t_n >$. Since $T \in \Omega^+$, then $D \in \Omega^+$, and so, by Lemma 5.1, $B \in \Omega^+$, and $Fv_2(B) = Fv_2(D) - \{C\} = Fv_2(T^+)$.

- (μ_g) : then $B = \mu C x_1...x_n D < t_1, ..., t_n >$, $T^+ = E[t_1/x_1, ..., t_n/x_n]$, and $D[E/C(x_1, ..., x_n)] \subseteq E$. Since $T^+ \in \Omega^+$, then $E \in \Omega^+$, and, by the induction hypothesis, $D[E/C(x_1, ..., x_n)] \in \Omega^+$, and $Fv_2(D[E/C(x_1, ..., x_n)]) \subseteq Fv_2(E)$. By Lemma 5.1, we have $D \in \Omega^+$, and $Fv_2(D) - \{C\} \subseteq Fv_2(D[E/C(x_1, ..., x_n)]) \subseteq Fv_2(E)$, and so $B \in \Omega^+$, and $Fv_2(B) = Fv_2(D) - \{C\} \subseteq Fv_2(D[E/C(x_1, ..., x_n)]) \subseteq Fv_2(E) = Fv_2(T^+)$. □

5.2 The TTR_0 type system

We define on the types of TTR a binary relation $\subseteq_0$ by the following way :
$A \subseteq_0$ B if and only if $A \subseteq B$, and in the proof we use only the weak version of $(\forall i_g)$:

$$(\forall i_{g_0})\ \frac{A[G/v] \subseteq_0 B}{\forall v A \subseteq_0 B}$$

where G is a term if v is an individual variable, and G is a predicate variable or a predicate symbol having the same arity of v if v is a predicate variable.

Lemma 5.2 *If $A \subseteq_0 B$, then, for every sequence of terms and/or formulas $\mathbf{G}$, $A[\mathbf{G}/\mathbf{v}] \subseteq_0 B[\mathbf{G}/\mathbf{v}]$, and we use the same proof rules.*

Proof Same proof as 1)-Lemma 4.2. □

Lemma 5.3 *Let A be an arrow type, and $Rep(A) = \forall \mathbf{v}(A_g \rightarrow A_d)$.*
1) If $A \in \Omega^-$ (resp. $A \in \Omega^+$), then $A_g \in \Omega^+$, and $A_d \in \Omega^-$ (resp. $A_g \in \Omega^-$, and $A_d \in \Omega^+$).
2) $A \subseteq_0 Rep(A)$, and $Rep(A) \subseteq_0 A$.

Proof By induction on A. □

Lemma 5.4 *If $T^- \in \Omega^-$, $T^+ \in \Omega^+$, and $T^- \subseteq T^+$, then $T^- \subseteq_0 T^+$.*

Proof By induction on the length of the derivation $T^- \subseteq T^+$. □

Definition We denote by TTR_0, the TTR type system whithout the rules (5), (7), (8) and by replacing the rule $(\subseteq)$ by :

$$(\subseteq_0)\ \frac{\Gamma \vdash_{TTR_0} t : A \quad A \subseteq_0 B}{\Gamma \vdash_{TTR_0} t : B}$$

Theorem 5.2 *Let $A_1, ..., A_n \in \Omega^-$, $\Gamma = x_1 : A_1, ..., x_n : A_n$, $A \in \Omega^+$, and t a normal λ-term. If $\Gamma \vdash_{TTR} t : A$, then $\Gamma \vdash_{TTR_0} t : A$, and in this typing each variable is assigned of a $\forall$-negative type, and each $u \in STE(t)$ is typable of a $\forall$-positive type.*

Proof We argue by induction on t.

- If $t = x_i$ $1 \leq i \leq n$, then $\forall \mathbf{v} A_i \subseteq A$, and $\mathbf{v}$ is not free in Γ. Since $A_i \in \Omega^-$, then $\forall \mathbf{v} A_i \in \Omega^-$, and, by Lemma 5.4, $\forall \mathbf{v} A_i \subseteq_0 A$. Therefore $\Gamma \vdash_{TTR_0} t : A$.

- If $t = \lambda x u$, then $\Gamma, x : B \vdash_{TTR} u : C$, $\forall \mathbf{v}(B \to C) \subseteq A$, and $\mathbf{v}$ is not free in Γ. Since $\forall \mathbf{v}(B \to C)$ is an arrow type, then, by Lemma 4.5, A is an arrow type. If $Rep(A) = \forall \mathbf{v}'(A_g \to A_d)$, then, by 1)-Lemma 5.3, $A_g \in \Omega^-$, and $A_d \in \Omega^+$. By Theorem 4.1, there is a sequence $\mathbf{G}$**,** such that $A_g \subseteq B[\mathbf{G}/\mathbf{v}]$, and $C[\mathbf{G}/\mathbf{v}] \subseteq A_d$. By 2)-Lemma 4.2, we have $\Gamma, x : B[\mathbf{G}/\mathbf{v}] \vdash_{TTR} u : C[\mathbf{G}/\mathbf{c}]$, and, by Lemma 4.9, $\Gamma, x : A_g \vdash_{TTR} u : A_d$. By the induction hypothesis, we have $\Gamma, x : A_g \vdash_{TTR_0} u : A_d$,and so, by 2)-Lemma 5.3, $\Gamma \vdash_{TTR_0} t : A$.

- If $t = (x_i)u_1...u_k$ $1 \leq i \leq n$ and $k \neq 0$, then $\forall \mathbf{v_0} A_i \subseteq C_1 \to B_1$, $\forall \mathbf{v_j} B_i \subseteq C_{j+1} \to B_{j+1}$ $1 \leq j \leq k-1$, $\forall \mathbf{v_k} B_k \subseteq A$ where $\mathbf{v_0}, ..., \mathbf{v_k}$ are not free in Γ, and $\Gamma \vdash_{TTR} u_j : C_j$ $1 \leq j \leq k$. By Theorems 4.1, 5.1, and Lemmas 4.4, 5.4, we have

  - $A_i = \forall \mathbf{v'_0} A'_i$, $A'_i = C'_1 \to \forall \mathbf{v'_1} B'_1$, $B_j = C'_{j+1} \to \forall \mathbf{v'_{j+1}} B'_{j+1}$ $1 \leq j \leq k-1$, $C'_j \in \Omega^+$, and $\forall \mathbf{v'_j} B'_j \in \Omega^-$ $1 \leq j \leq k$.
  - $C_j \subseteq C'_j[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{j-1}}/\mathbf{v'_{j-1}}]$, $\forall \mathbf{c'_j} B'_j[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{j-1}}/\mathbf{v'_{j-1}}] \subseteq \mathbf{B_j}$ $1 \leq j \leq k$, and $\forall \mathbf{v_k} \forall \mathbf{v'_k} B'_k[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{k-1}}/\mathbf{v'_{k-1}}] \subseteq_0 A$.

  Since $\Gamma \vdash_{TTR} u_j : C_j$ $1 \leq j \leq k$, then $\Gamma \vdash_{TTR} u_j : C'_j[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{j-1}}/\mathbf{v'_{j-1}}]$, and, by the induction hypothesis, $\Gamma \vdash_{TTR_0} u_j : C'_j[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{j-1}}/\mathbf{v'_{j-1}}]$. It is easy to check that $\Gamma \vdash_{TTR_0} t : B'_k[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{k-1}}/\mathbf{v'_{k-1}}]$, then
  $\Gamma \vdash_{TTR_0} t : \forall \mathbf{v_k} \forall \mathbf{v'_k} B'_k[\mathbf{G_0}/\mathbf{v'_0}]...[\mathbf{G_{k-1}}/\mathbf{v'_{k-1}}]$, and $\Gamma \vdash_{TTR_0} t : A$. □

6 Gődel transformation

6.1 $\perp$-types of TTR

Definition Let A be a type of TTR. We say that A is an $\perp$-type if and only if A is obtained by the following rules :

- $\perp$ is an $\perp$-type.

- If A is an $\perp$-type, then $B \to A$ is an $\perp$-type for every type B.

- If A is an $\perp$-type, then $\forall vA$ is an $\perp$-type for every variable v.

- If A is an $\perp$-type, C an n-ary predicate symbol which appears and is positive in A, $x_1, ..., x_n$ first order variables, and $t_1, ..., t_n$ terms, then $\mu C x_1 ... x_n A < t_1, ..., t_n >$ is an $\perp$-type.

Lemma 6.1 *If A is an $\perp$-type, and $A \subseteq B$, then B is an $\perp$-type.*

Proof By induction on the length of the derivation $A \subseteq B$. □

Lemma 6.2 *Let t be a normal λ-term, $A_1, ..., A_n \in \Omega^-$, $A \in \Omega^+$, $\perp$ does not appear in the types $A_1, ..., A_n, A$, and $B_1, ..., B_m$ are $\perp$-types. If $\Gamma = x_1 : A_1, ..., x_n : A_n, y_1 : B_1, ..., y_m : B_m \vdash_{TTR} t : A$, then $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} t : A$.*

Proof We argue by induction on t.

- If t is a variable, then $t = x_i$ $1 \leq i \leq n$ or $t = y_i$ $1 \leq i \leq m$.
  - The case $t = x_i$ is trivial.
  - If $t = y_i$, then $\forall \mathbf{v} B_i \subseteq A$ and $\mathbf{v}$ is not free in Γ. Since B_i is an $\perp$-type, then, by Lemma 6.1, A is an $\perp$-type, and $\perp$ appears in A. A contradictoire.

- If $t = \lambda x_{n+1} t'$, then $\Gamma, x_{n+1} : A_{n+1} \vdash_{TTR} t' : D$, $\forall \mathbf{v}(A_{n+1} \to D) \subseteq A$, $\mathbf{v}$ is not free in Γ. Since $A \in \Omega^+$, then, by Theorem 5.1, we have $A_{n+1} \in \Omega^-$, $D \in \Omega^+$, and $Fv_2(\forall \mathbf{v}(A_{n+1} \to D)) \subseteq Fv_2(A)$. Therefore $\perp$ does not appear in A_{n+1} and D. By the induction hypothesis, we have $x_1 : A_1, ..., x_n : A_n, x_{n+1} : A_{n+1} \vdash_{TTR} t : D$, and so $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} t : A$.

- If $t = (x)u_1 ... u_k$ $k \geq 1$, then two case can be see :
  - If $x = y_i$ $1 \leq i \leq m$, then, by Corollary 4.1, we have $\forall \mathbf{v_0} B_i \subseteq C_1 \to D_1$, $\forall \mathbf{v_j} D_j \subseteq C_{j+1} \to D_{j+1}$ $1 \leq i \leq k-1$, $\forall \mathbf{v_k} D_k \subseteq A$, where $\mathbf{v_0}, ..., \mathbf{v_k}$ are not free in A and Γ, and $\Gamma \vdash_{TTR} u_j : C_j$ $1 \leq j \leq k$. Since B_i is an $\perp$-type, then, by Lemma 6.1, D_j $1 \leq j \leq k$ and A are $\perp$-types, and $\perp$ appears in A. A contradictoire.
  - If $x = x_i$ $1 \leq i \leq n$, then, by Corollary 4.1, we have $\forall \mathbf{v_0} A_i \subseteq C_1 \to D_1$, $\forall \mathbf{v_j} D_j \subseteq C_{j+1} \to D_{j+1}$ $1 \leq j \leq k-1$, $\forall \mathbf{v_k} D_k \subseteq A$, where $\mathbf{v_0}, ..., \mathbf{v_k}$ are not free in A and Γ, and $\Gamma \vdash_{TTR} u_j : C_j$ $1 \leq j \leq k$. Since $A_i \in \Omega^-$, then, by Theorem 5.1, we have $C_j \in \Omega^+$, $D_i \in \Omega^-$ $1 \leq j \leq k$, and $Fv_2(C_j) \bigcup Fv_2(D_j) \subseteq Fv_2(A_i)$ $1 \leq j \leq k$. Therefore $\perp$ does not appear in C_j $1 \leq j \leq k$. By the inductive hypothesis, we have $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} u_j : C_j$ $1 \leq j \leq k$, and so $x_1 : A_1, ..., x_n : A_n \vdash_{TTR} t : A$. □

6.2 Gődel transformations

Definition With each predicate variable X, we associate a finite no empty set of predicate variables $V_X = \{X_1, ..., X_r\}$ having the same arity of X, such that : if $X \neq Y$, then $V_X \bigcap V_Y = \emptyset$. With each n-ary predicate variable X, and with each sequence of individual variables $x_1, ..., x_n$, we assosiate a formula F_X such that :

- F_X is an $\perp$-type ;

- F_X does not contain any predicate symbol ;

- the free variables of F_X are among $x_1, ..., x_n$ and the elements of V_X.

For each formula A, we define the formula A^* by the following induction way :

- If $A = C(t_1, ..., t_n)$, and C is a predicate symbol, then A^*=A.

- If $A = X(t_1, ..., t_n)$, and X is a predicate variable, then A^*=$F_X[t_1/x_1, ..., t_n/x_1]$.

- If $A = B \rightarrow C$, then A^*=$B^* \rightarrow C^*$.

- If $A = \forall xB$, then A^*=$\forall xB^*$.

- If $A = \forall XB$, then A^*=$\forall X_1 ... \forall X_r B^*$, where $V_X = \{X_1, ..., X_r\}$.

- If $A = \mu C x_1 ... x_n D < t_1, ..., t_n >$, then A^*=$\mu C x_1 ... x_n D^* < t_1, ..., t_n >$.

A^* is called the Gődel transformation of A.

Remark. In order to show that the above transformation is well defined, we need to prove the following Lemma :

Lemma 6.3 *Let C be a predicate variable or a predicate symbol, and A a type of TTR. If C is positive in A (resp. negative in A), then C is positive in A^* (resp. negative in A^*).*

Proof By induction on A. □

Lemma 6.4 *1) If $A \subseteq_0 B$, then $A^* \subseteq_0 B^*$, and we use the same proof rules.*
2) If $\Gamma \vdash_{TTR_0} t : A$, then $\Gamma^* \vdash_{TTR_0} t : A^*$, and we use the same typing rules.

Proof By induction on the length of the derivation $A \subseteq_0 B$ (resp. $\Gamma \vdash_{TTR_0} t : A$). □

Corollary 6.1 *Let $D \in \Omega^+$, and t a normal λ-term. If $\vdash_{TTR} t : D$, then $\vdash_{TTR} t : D^*$.*

Proof By induction on the length of the derivation $\vdash_{TTR} t : D$, and we use Theorem 5.2 and Lemma 6.4. □

7 Storage operators

7.1 Definition of storage operators

Definitions
1) Let T be a closed λ-term, and D, E two closed types of TTR (resp. $TTR^{\diamond}$). We say that T is a storage operator for the pair of types (D, E) if and only if for every λ-term t with $\vdash_{TTR} t : D$ (resp. $\vdash_{TTR^{\diamond}} t : D$), there are λ-terms τ_t and τ'_t such that $\tau_t \simeq_\beta \tau'_t$, $\vdash_{TTR} \tau'_t : E$ (resp. $\vdash_{TTR^{\diamond}} \tau'_t : E$), and for every $\theta_t \simeq_\beta t$, $(T)\theta_t f \succ (f)\tau_t[t_1/x_1, ..., t_n/x_n]$, where $Fv(\tau_t) = \{f, x_1, ..., x_n\}$ and $t_1, ..., t_n$ are λ-terms which depend on θ_t.
2) If $D = E$, we say that T is a storage operator for the type D.

Examples The type of recursive integers is the formula :

$$N^r[x] = \mu N x \Phi(N, x) < x >$$

where

$$\Phi(N, x) = \forall X\{\forall y(Ny \to Xsy), X0 \to Xx\}$$

(s is a unary function symbol for successor and 0 is a constant symbol for zero).
For each integer n, we define the recursive integer $\overline{n}$ by induction : $\overline{0} = \lambda f \lambda x x$ and $\overline{n+1} = \lambda f \lambda x (f)\overline{n}$. Let $\overline{N}$ be the set of recursive integers.
We have $\overline{N} = \{t \ / \ t$ is a closed normal λ-term / $\vdash_{TTR} t : N^r[s^n(0)]$, $n \geq 0\}$ (see [19]).
Let $\overline{s} = \lambda n \lambda f \lambda x (f) n$. It is easy to check that $\overline{s}$ is a λ-term for successor, and $\vdash_{TTR} \overline{s} : \forall y(N^r[y] \to N^r[sy])$.
Define
$T_1 = (Y)H$ where $H = \lambda x \lambda y((y)\lambda z(G)(x)z)\delta$, $G = \lambda x \lambda y(x)\lambda z(y)(\overline{s})z$, and $\delta = \lambda f(f)\overline{0}$;
$T_2 = \lambda \nu(\nu)\rho\tau\rho$ where $\tau = \lambda d \lambda f(f)\overline{0}$, and $\rho = \lambda y \lambda z(G)(y)z\tau z$,
then, for every $\theta_n \simeq_\beta \overline{n}$, $(T_i)\theta_n f \succ (f)(\overline{s})^n\overline{0}$ $(i = 1, 2)$.
Therefore, for every $n \geq 0$, T_1 and T_2 are storage operators for $N^r[s^n(0)]$.

Typing of T_1

We use in the typing the Gődel transformation with $V_X = \{X\}$, and $F_X = \neg X(x_1, ..., x_n)$ for every second order variable X of arity n.

- We have $\vdash_{TTR} \overline{0} : N^r[0]$, then $\vdash_{TTR} \delta : \neg\neg N^r[0]$.
- We have $\vdash_{TTR} \overline{s} : \forall y(N^r[y] \to N^r[sy])$, then
  $x : \neg\neg N^r[y], y : \neg N^r[sy], z : N^r[y] \vdash_{TTR} (y)(\overline{s})z : \bot$; hence :
  $x : \neg\neg N^r[y], y : \neg N^r[sy] \vdash_{TTR} (x)\lambda z(y)(\overline{s})z : \bot$; therefore :
  $\vdash_{TTR} G : \forall y(\neg\neg N^r[y] \to \neg\neg N^r[sy])$.

- We have $y : \Phi^*(N,x) \vdash_{TTR} y : \forall y(Ny \to \neg\neg N^r[sy]), \neg\neg N^r[0] \to \neg\neg N^r[x]$; thus :
  $x : \forall x(Nx \to \neg\neg N^r[x]), y : \Phi^*(N,x), z : Ny \vdash_{TTR} (G)(x)z : \neg\neg N^r[sy]$; therefore :
  $x : \forall x(Nx \to \neg\neg N^r[x]), y : \Phi^*(N,x) \vdash_{TTR} \lambda z(G)(x)z : \forall y(Ny \to \neg\neg N^r[sy])$; hence
  $x : \forall x(Nx \to \neg\neg N^r[x]) \vdash_{TTR} \lambda y((y)\lambda z(G)(x)z)\delta : \forall x(\Phi^*(N,x) \to \neg\neg N^r[x])$; therefore :
  $\vdash_{TTR} H : \forall x(Nx \to \neg\neg N^r[x]) \to \forall x(\Phi^*(N,x) \to \neg\neg N^r[x])$.

And finally $\vdash_{TTR} T_1 : \forall x\{N^r*[x] \to \neg\neg N^r[x]\}$.

Typing of T_2

We use in the typing the Gődel transformation with $V_X = \{X, X'\}$, and
$F_X = X(x_1, ..., x_n), X'(x_1, ..., x_n) \to \bot$ for every second order variable X of arity n.
Let $R = \forall X \forall y\{(X, X \to \neg\neg N^r[0], X \to \neg\neg N^r[y]), X \to \neg\neg N^r[sy]\}$, $D = R \to \neg\neg N^r[0]$, and $F[x] = R, D, R \to \neg\neg N^r[x]$.

- $\vdash_{TTR} \lambda f(f)\overline{0} : \neg\neg N^r[0]$; therefore : $\vdash_{TTR} \tau : X \to \neg\neg N^r[0]$, and $\vdash_{TTR} \tau : R \to \neg\neg N^r[0]$.
- By the previous typing, we have $\vdash_{TTR} G : \forall y(\neg\neg N^r[y] \to \neg\neg N^r[sy])$; hence :
  $y : X, X \to \neg\neg N^r[0], X \to \neg\neg N^r[y], z : X \vdash_{TTR} (G)(y)z\tau z : \neg\neg N^r[sy]$; therefore $\vdash_{TTR} \rho : R$.
- Check that $\Phi^*(\lambda x F[x]/N, x) \subseteq F[x]$.
  $\Phi^*(\lambda x F[x]/N, x) =$
  $\forall X \forall X'\{\forall y(F[y], Xsy, X'sy \to \bot), (X0, X'0 \to \bot) \to (Xx, X'x \to \bot)\}$;
  therefore by specifying Xx by R, and $X'x$ by $\neg N^r[x]$; we obtain :
  $\Phi^*(\lambda x F[x]/N, x) \subseteq \forall y(F[y], R, \neg N^r[sy] \to \bot), (R, \neg N^r[0] \to \bot) \to (R, \neg N^r[x] \to \bot)$. We need to check that $R \subseteq \forall y(F[y], R, \neg N^r[sy] \to \bot)$, this is absolutely true.

Therefore $N^r*[x] \subseteq F[x]$ and $\nu : N^r*[x] \vdash_{TTR} \nu : R, D, R \to \neg\neg N^r[x]$; then :
$\nu : N^r*[x] \vdash_{TTR} (\nu)\rho\tau\rho : \neg\neg N^r[x]$; and finally $\vdash_{TTR} T_2 : \forall x\{N^r*[x] \to \neg\neg N^r[x]\}$.

7.2 General Theorem

Theorem 7.1 *Let D, E be two $\forall$-positive closed types of TTR, such that $\bot$ does not appear in E. If $\vdash_{TTR} T : D* \to \neg\neg E$, then T is a storage operator for the pair (D, E).*

Proof It is a consequence from the following Theorem :

Theorem 7.2 *Let D, E be two $\forall$-positive closed types of $TTR^\diamond$, such that $\bot$ does not appear in E. If $\vdash_{TTR} T : D* \to \neg\neg E$, then T is a storage operator for the pair (D, E).*

Indeed:

Lemma 7.1 *1) If $T \in \Omega^+$ (resp. $T \in \Omega^-$) then $T^\diamond \in \Omega^+$ (resp. $T^\diamond \in \Omega^-$).*
*2) For each Gődel transformation * of TTR, there is a Gődel transformation $*'$ of $TTR^\diamond$ such that : for every type D of TTR, $D^{*\diamond} = D^{\diamond *'}$.*

Proof 1) By induction on T.
2) $*'$ is the restiction of * on the types of $TTR^\diamond$. □

Let t be a normal λ-term, such that $\vdash_{TTR} t : D$. If $\vdash_{TTR} T : D^* \rightarrow \neg\neg E$, then, by Theorem 3.3, $\vdash_{TTR^\diamond} T : D^{*\diamond} \rightarrow \neg\neg E^\diamond$. By 2)-Lemma 7.1, there is a Gődel transformation $*'$, such that $\vdash_{TTR^\diamond} T : D^{\diamond *'} \rightarrow \neg\neg E^\diamond$. Therefore, there are λ-terms τ_t and τ'_t, such that $\tau_t \simeq_\beta \tau'_t$, $\vdash_{TTR^\diamond} \tau'_t : E^\diamond$, and $(T)tf \succ (f)\tau_t[t_1/x_1, ..., t_n/x_n]$. By 2)-Corollary 6.1, we have $\vdash_{TTR} t : D^*$, then $f : \neg E \vdash_{TTR} (T)tf :\bot$, and $f : \neg E \vdash_{TTR} (f)\tau_t[t_1/x_1, ..., t_n/x_n] :\bot$. Therefore $f : \neg E \vdash_{TTR} (f)\tau'_t :\bot$, and, by Corollary 4.1, $\vdash_{TTR} \tau'_t : E$. □

We give the proof of Theorem 7.2 in a particular case.

Let $N^r = \mu N[\forall X\{N \rightarrow X, X \rightarrow X\}]$, and * the Gődel transformation with $V_X = \{X\}$, and $F_X = \neg X(x_1, ..., x_n)$ for every second order variable X of arity n.
We will prove that : If $\vdash_{TTR^\diamond} T : N^{r*} \rightarrow \neg\neg N^r$, then T is a storage operator for N^r.
Because of : if t is a closed normal λ-term with $\vdash_{TTR^\diamond} t : N^r$, then $t = \overline{n}$ for a certain integer n, and it is suffies to prove that : If $\vdash_{TTR^\diamond} T : N^{r*} \rightarrow \neg\neg N^r$, then, for every $n \geq 0$, there is an $m \geq 0$ and $\tau \simeq_\beta \overline{m}$, such that, for every λ-term $\theta_n \simeq_\beta \overline{n}$, there is a substitution σ, such that $(T)\theta_n f \sim (f)\sigma(\tau)$.

Lemma 7.2 *If $\Gamma' = \Gamma, x : N^{r*} \vdash_{TTR^\diamond} (x)u_1...u_n :\bot$, then $n = 3$, and there is a type G, such that $\Gamma' \vdash_{TTR^\diamond} u_1 : N^{r*} \rightarrow \neg G$, $\Gamma' \vdash_{TTR^\diamond} u_2 : \neg G$, and $\Gamma' \vdash_{TTR^\diamond} u_3 : G$.*

Proof By Corollary 4.1, we have $\forall \mathbf{v_0} N^{r*} \subseteq A_1 \rightarrow B_1$, $\forall \mathbf{v_i} B_i \subseteq A_{i+1} \rightarrow B_{i+1}$ $1 \leq i \leq n-1$, $\forall \mathbf{v_n} B_n \subseteq \bot$, $\mathbf{v_0}, ..., \mathbf{v_n}$ are not free in N^{r*} and Γ, and $\Gamma' \vdash_{TTR^\diamond} u_i : A_i$ $1 \leq i \leq n$. Since $\forall \mathbf{v_0} N^{r*} \subseteq A_1 \rightarrow B_1$, then, by Theorem 4.1, there is a formula F, such that $A_1 \subseteq N^{r*} \rightarrow \neg F$ and $\neg F \rightarrow \neg F \subseteq B_1$. We have also $\forall \mathbf{v_1} B_1 \subseteq A_2 \rightarrow B_2$, then $\forall \mathbf{v_1}(\neg F \rightarrow \neg F) \subseteq A_2 \rightarrow B_2$, and, by Theorem 4.1, there is a sequence of formulas $\mathbf{F_1}$, such that $A_2 \subseteq \neg F[\mathbf{F_1}/\mathbf{v_1}]$ and $\neg F[\mathbf{F_1}/\mathbf{v_1}] \subseteq B_2$. Now, since $\forall \mathbf{v_2} B_2 \subseteq A_3 \rightarrow B_3$, we have $\forall \mathbf{v_2}(\neg F[\mathbf{F_1}/\mathbf{v_1}]) \subseteq A_3 \rightarrow B_3$, and, by Theorem 4.1, there is a sequence of formulas $\mathbf{F_2}$, such that $A_3 \subseteq F[\mathbf{F_1}/\mathbf{v_1}, \mathbf{F_2}/\mathbf{v_2}]$ and $\bot \subseteq B_3$. By Corollary 4.1, we have $n = 3$. Let $G = F[\mathbf{F_1}/\mathbf{v_1}, \mathbf{F_2}/\mathbf{v_2}]$. Since $\mathbf{v_1}, \mathbf{v_2}$ are not free in N^{r*} and Γ, we deduce $\Gamma' \vdash_{TTR^\diamond} u_1 : N^{r*} \rightarrow \neg G$, $\Gamma' \vdash_{TTR^\diamond} u_2 : \neg G$, and $\Gamma' \vdash_{TTR^\diamond} u_3 : G$. □

Let $n \geq 0$.

Definition An n-special application θ is a function from $\{0,1,...,n\}$ to Λ with the following properties : $\theta(0) \succ \overline{0}$ and $\theta(m+1) \succ \lambda f_m \lambda x_m (f_m)\theta(m)$ $0 \leq m \leq n-1$.

Lemma 7.3 *For every $\theta_n \simeq_\beta \overline{n}$, there is an n-special application θ, such that $\theta(n) = \theta_n$.*

Proof Easy. □

Definitions
1) Let $0 \leq m \leq n$ and $\mathbf{u}= u_{m,1}, u_{m,2}, u_{m,3}, ..., u_{n-1,1}, u_{n-1,2}, u_{n-1,3}$ a sequence of λ-terms. We denoted by $x_{m,\mathbf{u}}$ a constant which does not appear in $\mathbf{u}$.
2) Let θ be an n-special application. The n-special substitution S_θ is the function on the set Λ defined by induction :

- If $u = x$, then $S_\theta(x) = x$;

- If $u = \lambda x v$, then $S_\theta(u) = \lambda y S_\theta(v[y/x])$ where $y \notin Fv(\theta(n))$;

- If $u = (v)w$, then $S_\theta(u) = (S_\theta(v))S_\theta(w)$;

- If $u = x_{m,\mathbf{u}}$,then
$S_\theta(u) = \theta(m)[S_\theta(u_{m,1})/f_m, S_\theta(u_{m,2})/x_m, ..., S_\theta(u_{n-1,1})/f_{n-1}, S_\theta(u_{n-1,2})/x_{n-1}]$.

An n-special substitution is the application S_θ associated to a some n-special application θ.

Lemma 7.4 *Let $\{U_i \succ V_i\}_{1 \leq i \leq r}$ be a sequence of head reductions such that :*
$V_i = (x_{m,\mathbf{u}})u_1u_2u_3$ $0 \leq m \leq n$, $[U_{i+1} = (u_1)x_{m-1,u_1,u_2,u_3,\mathbf{u}}u_3$ if $m \neq 0$, and $U_{i+1} = (u_2)u_3$ if $m = 0]$, and S_θ an n-special substitution. For every $1 \leq i \leq r$, $S_\theta(U_1) \sim S_\theta(V_i)$.

Proof We argue by induction on i.
The case $i = 0$ is a consequence of Theorem 2.1.
Assume that is true for i, and prove it for $i+1$.
If $V_i = (x_{m,\mathbf{u}})u_1u_2u_3$ $0 \leq m \leq n$, then
$S_\theta(V_i) = (\theta(m)[S_\theta(u_{m,1})/f_m, S_\theta(u_{m,2}/)x_m, ..., S_\theta(u_{n-1,1})/f_{n-1}, S_\theta(u_{n-1,2})/x_{n-1}])$
$S_\theta(u_1)S_\theta(u_2)S_\theta(u_3)$.

- If $m \neq 0$, then $\theta(m) \succ \lambda f_{m-1} \lambda x_{m-1}(f_{m-1})\theta(m-1)$,
and $S_\theta(V_i) \sim (S_\theta(u_1))\theta(m-1)[S_\theta(u_{m-1,1})/f_{m-1}, S_\theta(u_{m-1,2})/x_{m-1}, ...,$
$S_\theta(u_{n-1,1})/f_{n-1}, S_\theta(u_{n-1,2})/x_{n-1}])S_\theta(u_3) = S_\theta(U_{i+1})$.

- If $m = 0$, then $\theta(m) \succ \lambda f \lambda x x$, and $S_\theta(V_i) \sim (\lambda f \lambda x x)S_\theta(u_1)S_\theta(u_2)S_\theta(u_3) \sim (S_\theta(u_2))S_\theta(u_3) = S_\theta(U_{i+1})$.

By the induction hypothesis we have $S_\theta(U_1) \sim S_\theta(V_i)$, then $S_\theta(U_1) \sim S_\theta(U_{i+1})$, and, by Theorem 2.1, $S_\theta(U_1) \sim S_\theta(V_{i+1})$. □

Definition A context $\Gamma = f : \neg N, x_{n,\mathbf{u_0}} : N^r*, x_{m_1,\mathbf{u_1}} : N^r*, ..., x_{m_s,\mathbf{u_s}} : N^r*$ where $0 \leq m_j \leq n$, $1 \leq j \leq s$, is called n-good.

Lemma 7.5 *There is a sequence of head reductions $\{U_i \succ V_i\}_{1 \leq i \leq r}$ such that :*

- *$U_1 = (T)x_n f$ and $V_r = (f)\tau$ where $\tau \simeq_\beta \overline{l}$ for some $l \geq 0$;*

- *$V_i = (x_{m,\mathbf{u}})u_1u_2u_3$ $0 \leq m \leq n$, and*
$U_{i+1} = (u_1)x_{m-1,u_1,u_2,u_3,\mathbf{u}}u_3$ if $m \neq 0$, and $U_{i+1} = (u_2)u_3$ if $m = 0$;

- *For every $1 \leq i \leq r$, there is an n-good context Γ_i such that $\Gamma_i \vdash_{TTR^\diamond} V_i : \bot$.*

Proof Since $\vdash_{TTR^\diamond} T : N^r* \rightarrow \neg\neg N^r$, then $x_n : N^r*, f : \neg N^r \vdash_{TTR^\diamond} (T)x_n f : \bot$, and, by Corollary 4.3 and Lemma 7.2, we have $(T)x_n f \succ V_1$ where $V_1 = (f)\tau$ or $V_1 = (x_n)u_1u_2u_3$.
Assume that we have the head reduction $U_k \succ V_k$ and $V_k \neq (f)\tau$. Then $V_k = (x_{m,\mathbf{u}})u_1u_2u_3$ $0 \leq m \leq n$, and, by the induction hypothesis, there is an n-good context Γ_k such that $\Gamma_k \vdash_{TTR^\diamond} (x_{m,\mathbf{u}})u_1u_2u_3 : \bot$. By Lemma 7.2, there is a type G, such that $\Gamma_k \vdash_{TTR^\diamond} u_1 : N^r* \rightarrow \neg G$, $\Gamma_k \vdash_{TTR^\diamond} u_2 : \neg G$, and $\Gamma_k \vdash_{TTR^\diamond} u_3 : G$.

- If $m = 0$, let $U_{k+1} = (u_2)u_3$. Let $\Gamma_{k+1} = \Gamma_k$. We have $\Gamma_{k+1} \vdash_{TTR^\diamond} U_k : \bot$.

- If $m \neq 0$, let $U_{k+1} = (u_1)x_{m-1,u_1,u_2,u_3,\mathbf{u}}u_3$. The variable $x_{m-1,u_1,u_2,u_3,\mathbf{u}}$ is not used before. Indeed, if it is, by Lemma 7.4, the λ-term $(T)\overline{n}f$ is not solvable. That is impossible because $f : \neg N^r \vdash_{TTR^\diamond} (T)\overline{n}f : \bot$. Therefore $\Gamma_{k+1} = \Gamma_k, x_{m-1,u_1,u_2,u_3,\mathbf{u}} : N^r*$ is an n-good context and $\Gamma_{k+1} \vdash_{TTR^\diamond} U_{k+1} : \bot$.

By Corollary 4.3 and Lemma 7.2, we have $U_{k+1} \succ V_{k+1}$ where $V_{k+1} = (f)\tau$ or $V_{k+1} = (x_{s,\mathbf{v}})v_1v_2v_3$ $0 \leq s \leq n$.
This constraction always terminates. Indeed, if not, by Lemma 7.4, the λ-term $(T)\overline{n}f$ is not solvable. That is impossible because $f : \neg N^r \vdash_{TTR^\diamond} (T)\overline{n}f : \bot$.
Therefore there is $r \geq 0$ and an n-good context Γ_r such that $V_r = (f)\tau$ and $\Gamma_r \vdash_{TTR^\diamond} V_r : \bot$. By Lemma 6.2, we have $\tau \simeq_\beta \overline{l}$ for some $l \geq 0$. □

Let θ_n be a λ-term such that $\theta_n \simeq_\beta \overline{n}$. By Lemma 7.3, let θ be an n-special application such that $\theta(n) = \theta_n$. Let S_θ the n-special substitution associated to θ. By Lemma 7.4, we have for every $1 \leq i \leq r$, $(T)\theta_n f \sim S_\theta(V_i)$. In particular, for $i = n$, $(T)\theta_n f \sim S_\theta((f)\tau) = (f)S_\theta(\tau)$. Then T is a storage operator for N^r. □

References

[1] H. BARENDREGT. *The lambda calculus: Its Syntax and Semantics.*
North Holland, 1984.

[2] R. DAVID. *The Inf function in system F.*
Theorical Computer Science, 135 (423-431), 1994.

[3] P. GIANNINI and S. RONCHI. *Characterization of typing in polymorphic type discipline.*
LICS, Edinboug (61-70), 1988.

[4] J.L. KRIVINE. *Lambda calcul, évaluation paresseuse et mise en mémoire.*
Informatique Théorique et Applications, Vol. 25,1, p. 67-84, 1991.

[5] J.L. KRIVINE. *Lambda calcul, types et modèle.*
Masson, Paris, 1990.

[6] J.L. KRIVINE. *Opérateurs de mise en mémoire et traduction de Gődel.*
Archive. Math. Logic 30. (241-267), 1990.

[7] J.L. KRIVINE. *Mise en mémoire (preuve générale).*
Manuscript,1991.

[8] R. LABIB-SAMI. *Typer avec (ou sans) types auxilières.*
Manuscript, 1986.

[9] D. LEIVANT. *Reasonning about functional programs and complexity classes associated with type disciplines.*
In 24th Annual Symposium on Foundations of Computer Science, volum 44 (460-469), 1983.

[10] D. LEIVANT. *Typing and computation properties of lambda expressions.*
Theorical Computer Science, 44 (51-68), 1986.

[11] J. MITCHELL. *Polomorphic type.*
Information and Computation, 76 (2/3), (211-249), 1988.

[12] K. NOUR. *Opérateurs de mise en mémoire en lambda-calcul pur et typé.*
Thèse de doctorat, Université de Savoie, 1993.

[13] K. NOUR. *Strong storage operators and data types.*
Archive. Math. Logic 34. (65-78), 1995.

[14] K. NOUR. *Opérateurs propres de mise en mémoire.*
C.R.A.S. Paris, t. 317, Série I, p. 1-6, 1993.

[15] K. NOUR. *Preuve syntaxique d'un théorème de J.L. Krivine sur les opérateurs de mise en mémoire.*
C.R.A.S. Paris, t. 318, Série I, p. 201-204, 1994.

[16] K. NOUR. *Opérateurs de mise en mémoire et types $\forall$-positifs.*
Submitted to publication in Thearetical Informatics and Applications, 1993.

[17] K. NOUR and R. DAVID. *Storage operators and directed lambda-calculus.*
Journal of Symbolic Logic (to appear).

[18] M. PARIGOT. *Programming with proofs : a second order type theory.*
ESOP'88, LNCS 300, (145-159), 1988.

[19] M. PARIGOT. *On representation of data in lambda calculus.*
To appear in LNCS.

[20] M. PARIGOT. *Recursive programming with proofs.*
Theoritical Computer Science, 94 (335-356), 1992.